FLEXBOX

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University
https://softuni.bg

Table of Contents

Have a Question?

Software University

sli.do

#HTML-CSS

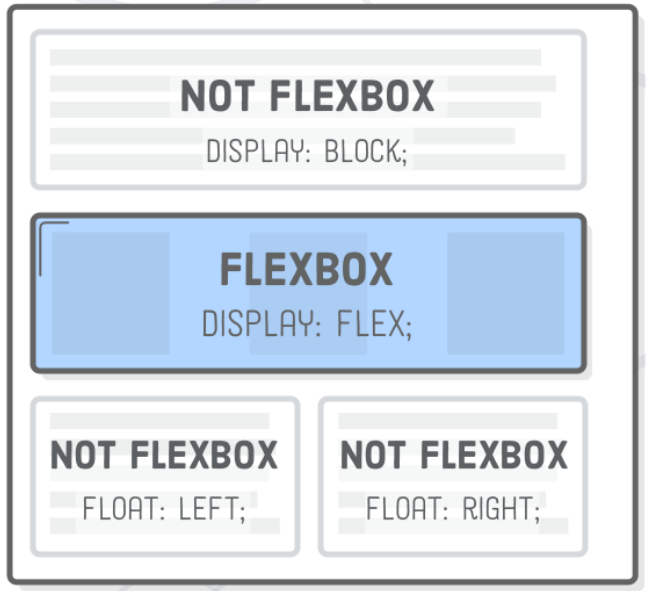
FLEXBOX

What is Flexbox?

- The Flexible Box Module - **flexbox**, was designed as a one-dimensional layout model, and as a method that could offer **space distribution** between items in an interface and powerful alignment capabilities
- Flexbox is a method for laying out items in **rows** or **columns**
- Items flex to **fill** additional space and **shrink** to fit into smaller spaces

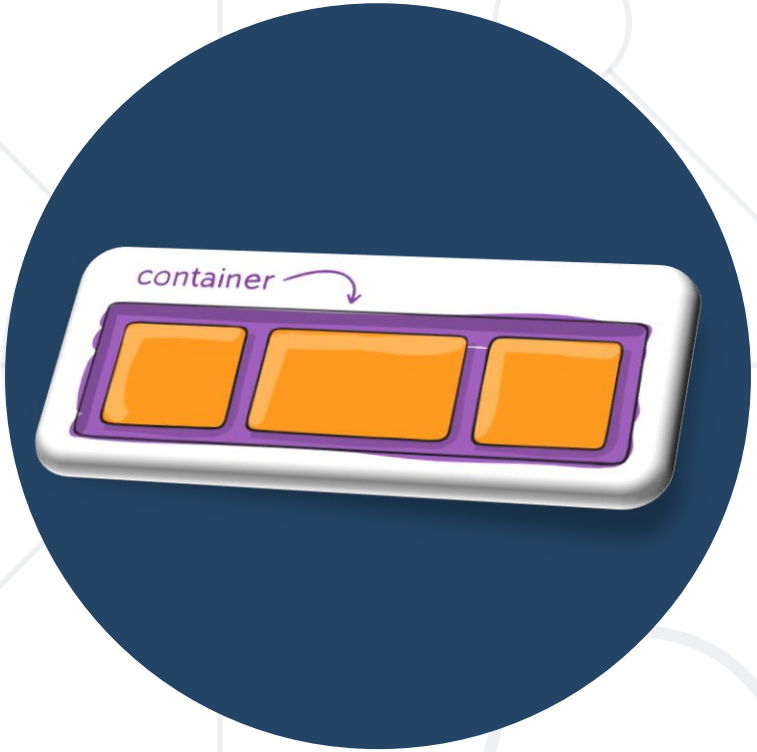
Why Flexbox?

- For a long time, the only reliable cross browser-compatible tools available for creating CSS layouts were **floats** and **positioning**
- These are fine and they work, but in some ways, they are also rather limiting and frustrating



Why Flexbox?

- The following simple layout requirements are either **difficult** or **impossible** to achieve with such tools, in any kind of convenient, flexible way:
  - Vertically centering a block of content inside its parent
  - Making all the children of a container take up an equal amount of the available width/height, regardless of how much width/height is available
  - Making all columns in a multiple column layout adopt the same height even if they contain a different amount of content



Properties for the Parent

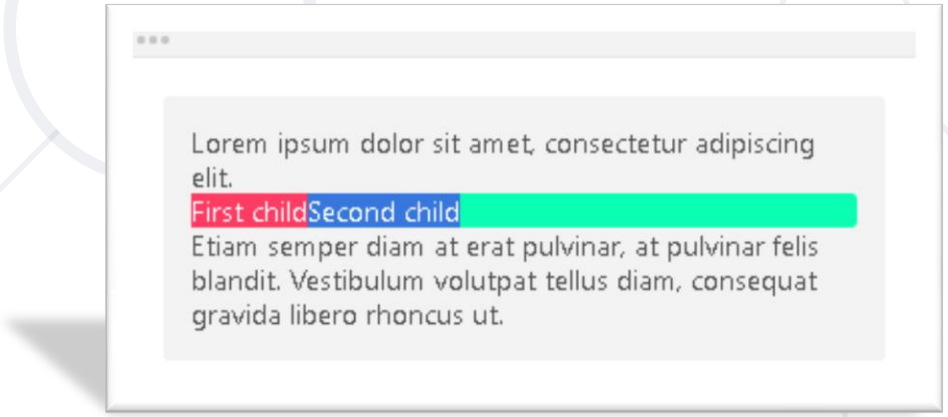
(flex container)

Display - flex

- The element is turned into a **flexbox** container. On its own, it behaves like a block element
- Its child elements will be turned into **flexbox items**

```
<body>
    <p>Lorem ipsum dolor sit amet, co
nsectetur adipiscing elit.</p>
    <div class="container">
        <p>First child</p>
        <p>Second child</p>
    </div>
    <p>Etiam semper diam at erat pulv
inar, at pulvinar felis blandit. Vest
ibulum volutpat tellus diam, consequa
t gravida libero rhoncus ut.
    </p>
</body>
```

```
.container {
    display: flex;
}
```

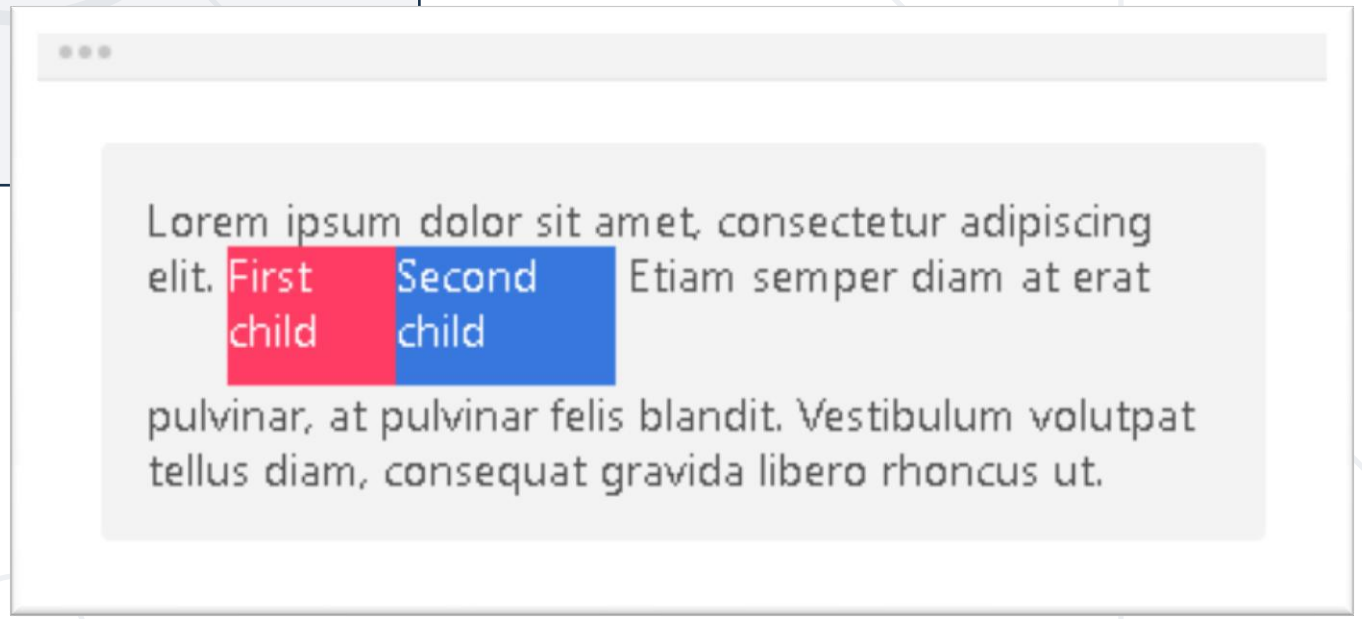


Display – inline-flex

- The element shares properties of both an **inline** and a **flexbox** element:
  - **inline** because the element behaves like simple text, and inserts itself in a block of text
  - **flexbox** because its child element will be turned into flexbox items

Display – inline-flex Example

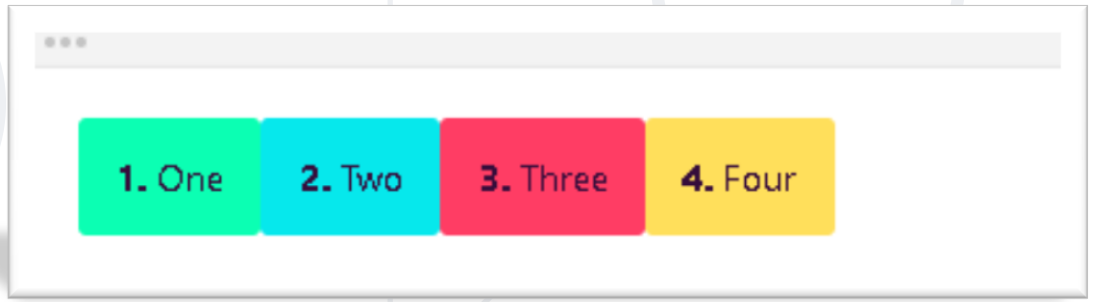
```
.container {
  display: inline-flex;
  height: 3em;
  width: 120px;
}
```



Flex-direction

- Defines how flexbox items are ordered within a flexbox container
  - **flex-direction: row;**

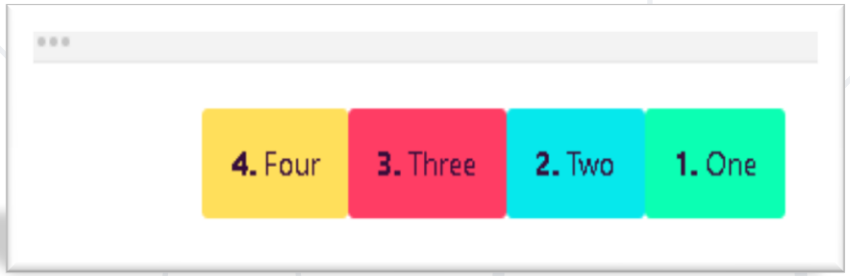
  The flexbox items are ordered the **same way** as the text direction, along the main axis



Flex-direction

- **flex-direction: row-reverse;**

The flexbox items are ordered the **opposite** way as the **text direction**, along the main axis



- **flex-direction: column;**

The flexbox items are ordered the **same** way as the **text direction**, along the **cross axis**

Flex-direction

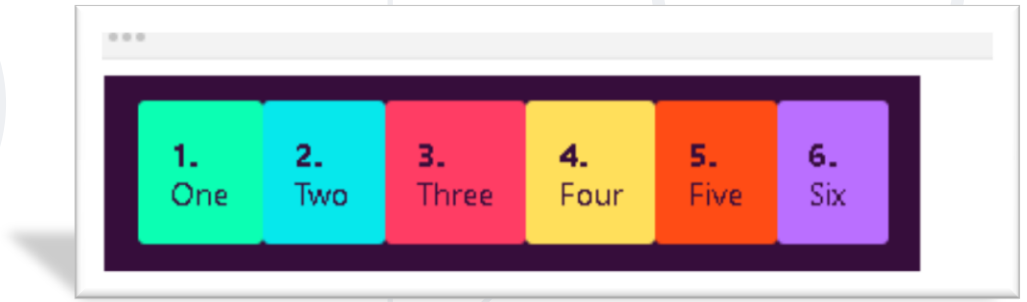
- **flex-direction: column-reverse;**

The flexbox items are ordered the **opposite** way as the **text direction**, along the **cross axis**

Flex-wrap

- Defines if flexbox items appear on a **single line** or on **multiple lines** within a flexbox container
  - **flex-wrap: nowrap;**

  The flexbox items will remain on a **single line**, no matter what, and will eventually overflow if needed



Flex-wrap

- **flex-wrap: wrap;**

The flexbox items will be distributed among **multiple lines** if needed

- **flex-wrap: wrap-reverse;**

The flexbox items will be distributed among **multiple lines** if needed. Any additional line will appear **before** the previous one

Flex-flow

- **Flex-flow** is a shorthand for the flex-direction and flex-wrap properties
- The default value is **row nowrap**

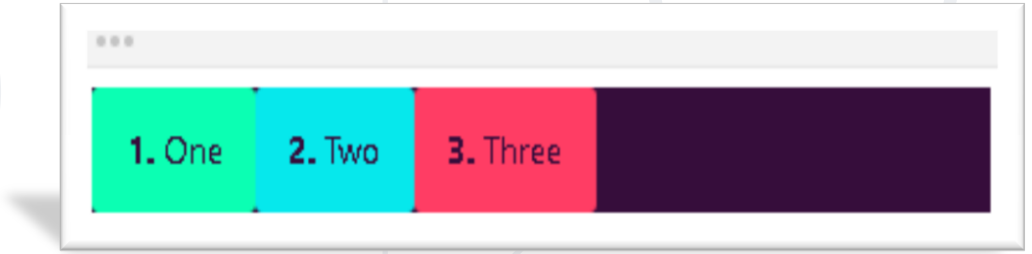
```
flex-flow: <flex-direction> || <flex-wrap>

.container {
    flex-flow: row wrap;
}
```

Justify-content

- Justify-content: Defines how flexbox/grid items are aligned according to the **main** axis, within a flexbox container
  - **justify-content: flex-start;**

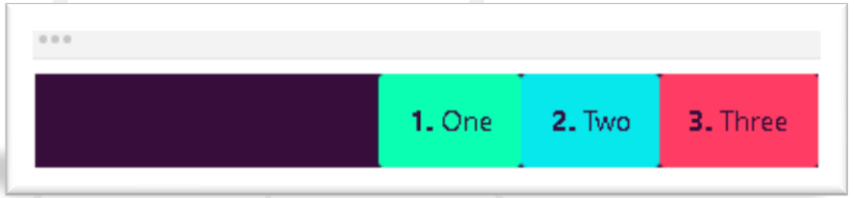
  The flexbox items are pushed towards the **start** of the container's main axis



Justify-content

- **justify-content: flex-end;**

The flexbox items are pushed towards the **end** of the container's main axis



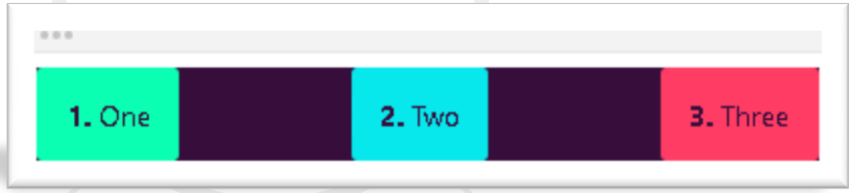
- **justify-content: center;**

The flexbox items are **centered** along the container's main axis

Justify-content

- **justify-content: space-between;**

The remaining space is distributed **between** the flexbox items



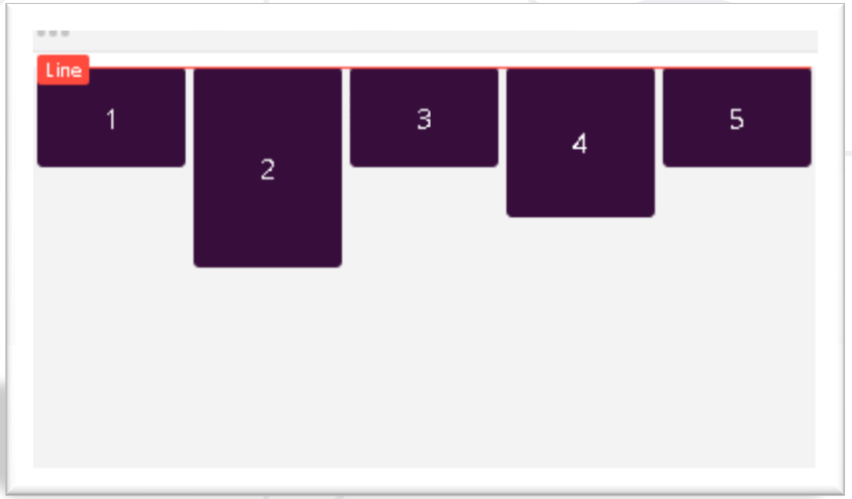
- **justify-content: space-around;**

The remaining space is distributed **around** the flexbox items: this adds space ***before*** the first item and ***after*** the last one

Align-items

- Align-items: Defines how flexbox items are aligned according to the **cross** axis, within a line of a flexbox container
  - **align-items: flex-start;**

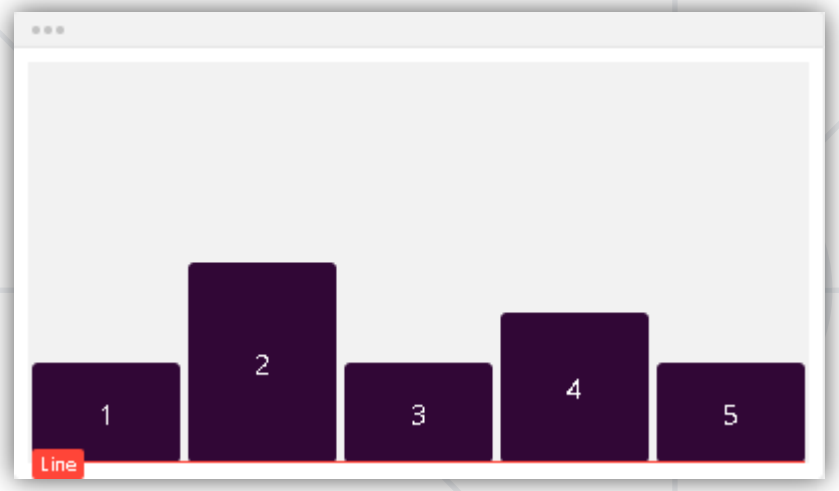
  The flexbox items are aligned at the **start** of the **cross axis**



Align-items

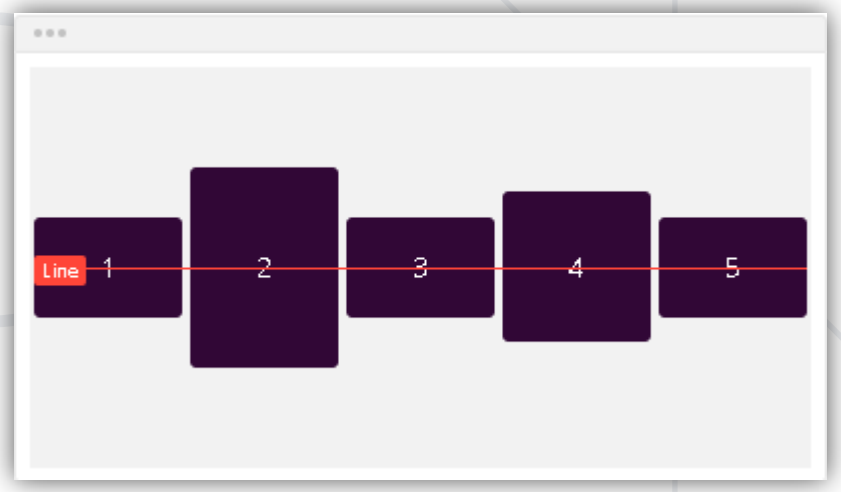
- **align-items: flex-end;**

The flexbox items are aligned at the **end** of the **cross axis**



- **align-items: center;**

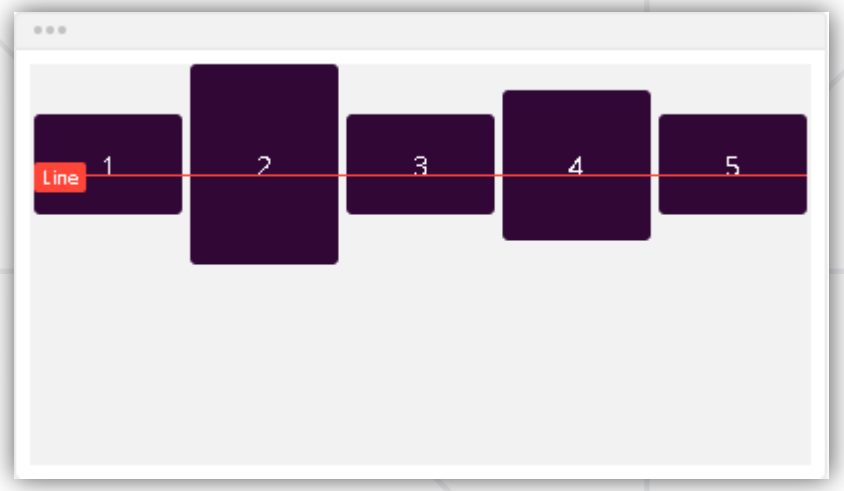
The flexbox items are aligned at the **center** of the **cross axis**



Align-items

- **align-items: baseline;**

The flexbox items are aligned at the **baseline** of the **cross axis**



- **align-items: stretch;**

The flexbox items will stretch across the whole **cross axis**

Align-content

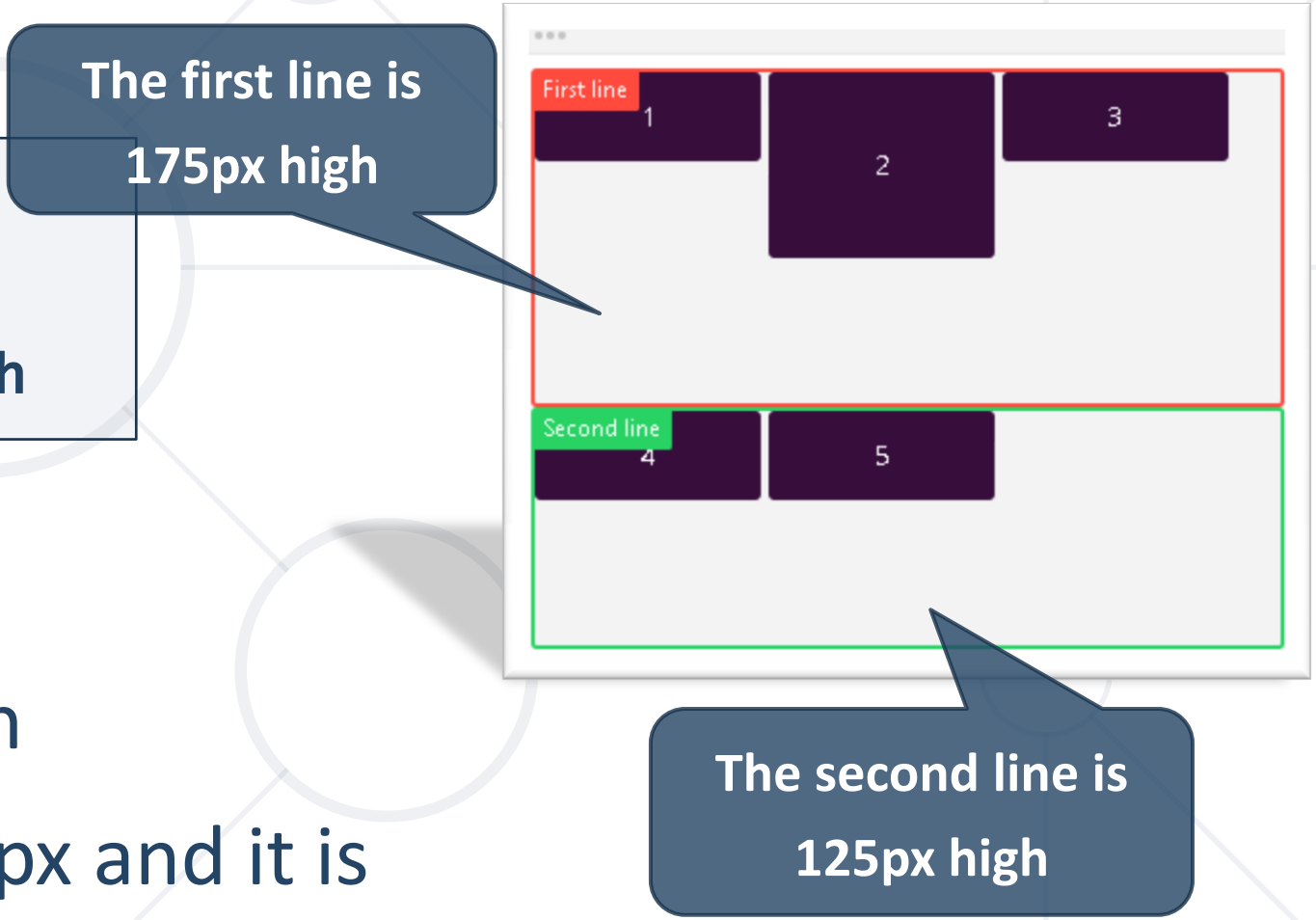
- Align-content: Defines how each line is aligned within a flexbox container
- It only applies if flex-wrap: wrap is present, and if there are multiple lines of flexbox items
  - **align-content: stretch;**

  Each line will stretch to ***fill*** the remaining space

Align-content: stretch Example

- Example:

```
The container is 300px high
All boxes are 50px high
The second box is 100px high
```

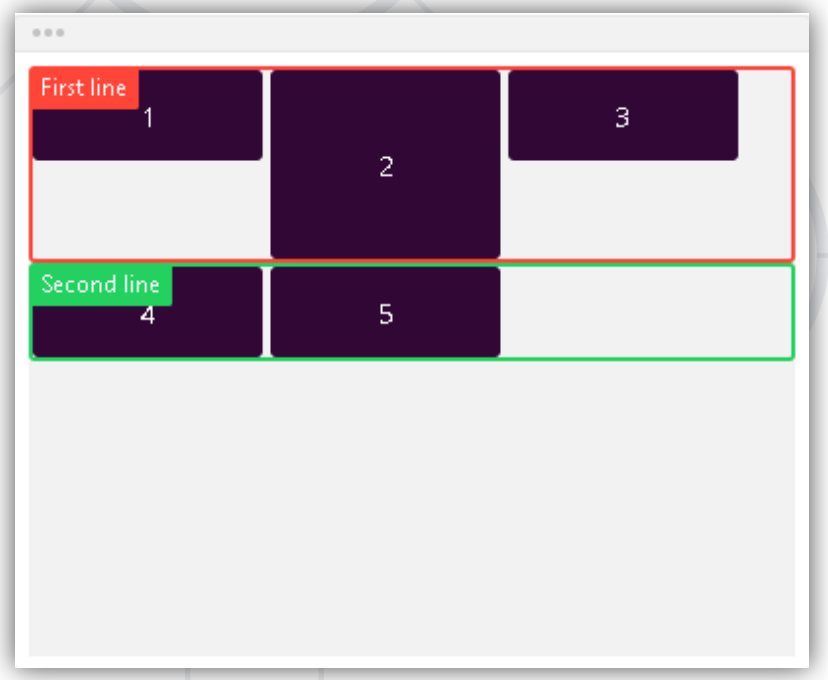


- The first line is 100px high
- The second line is 50px high
- The remaining space is 150px and it is distributed equally amongst the two lines

Align-content

- **align-content: flex-start;**

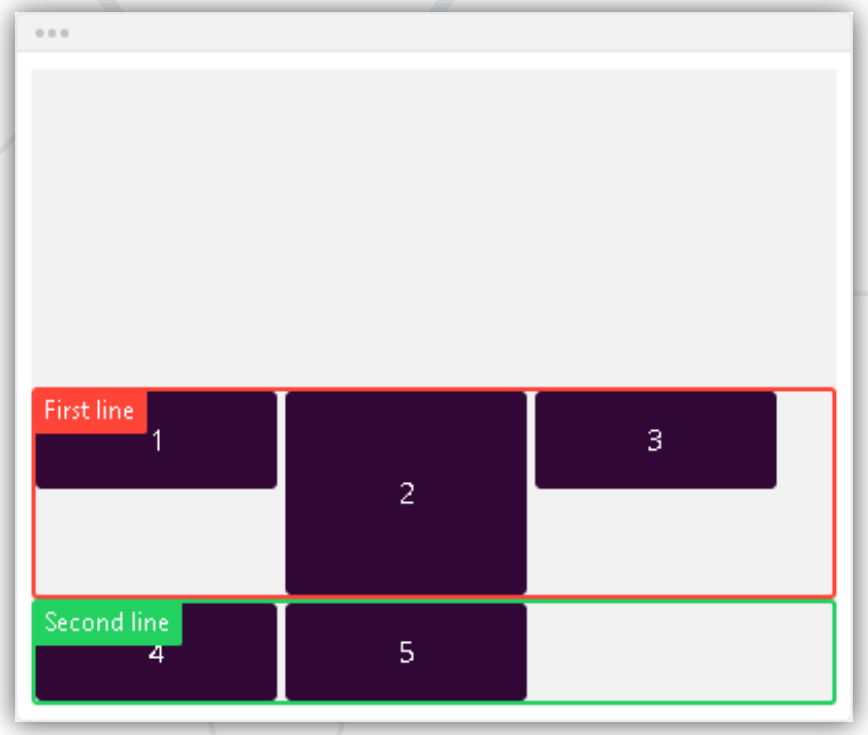
Each line will only fill the space it *needs*. They will all move towards the **start** of the flexbox container's cross axis



Align-content

- **align-content: flex-end;**

Each line will only fill the space it *needs*. They will all move towards the **end** of the flexbox container's cross axis



Align-content

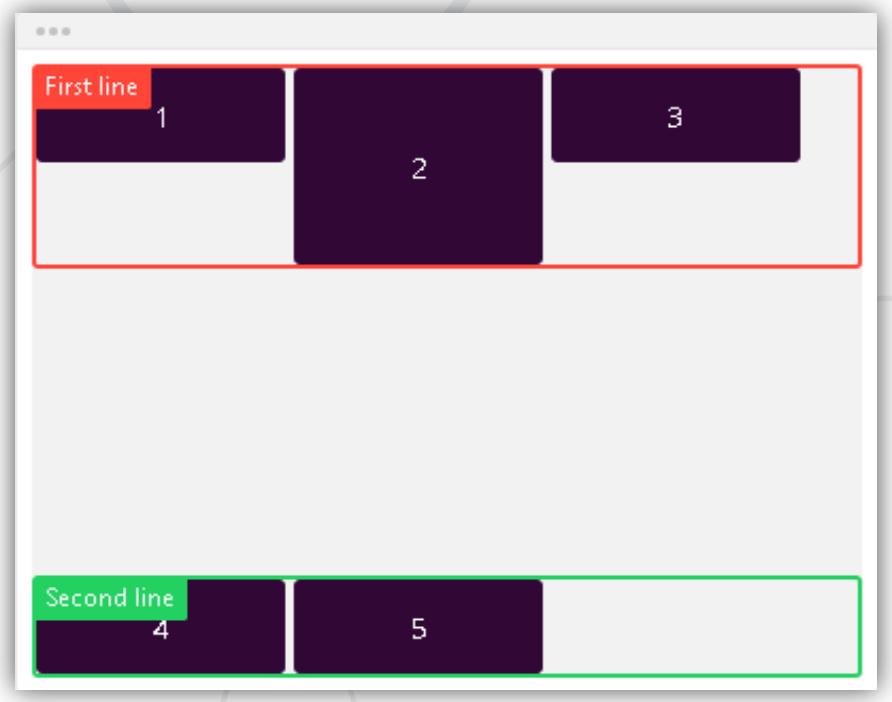
- **align-content: center;**

Each line will only fill the space it *needs*. They will all move towards the **center** of the flexbox container's cross axis

Align-content

- **align-content: space-between;**

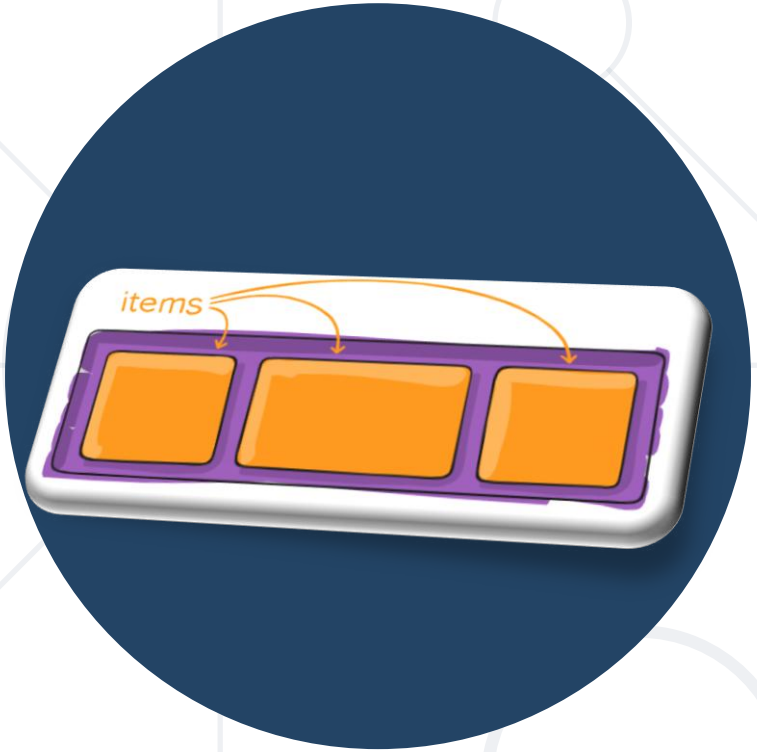
Each line will only fill the space it *needs*. The *remaining* space will appear **between** the lines



Align-content

- **align-content: space-around;**

Each line will only fill the space it *needs*. The **remaining** space will be distributed equally **around** the lines: before the first line, between the two, and after the last one



Properties for the Children

(flex items)

Order

- Order - defines the order of a flexbox item
  - **order: 0;**

The order of the flexbox items is the one defined in the **HTML code**

Order

- **order: 1;**

The order is **relative** to the flexbox item's **siblings**. The final order is defined when all individual flexbox item order values are taken into account

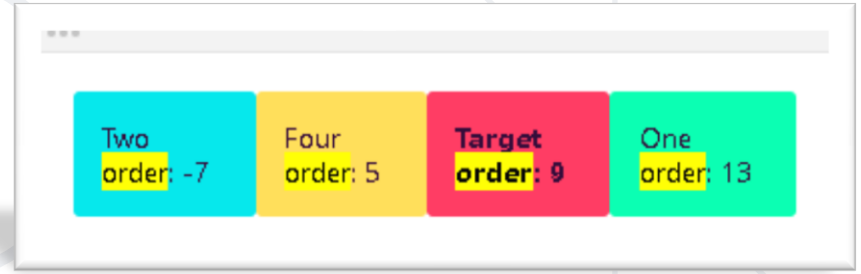
Order

- **order: -1;**

You can use **negative** values

- **order: 9;**

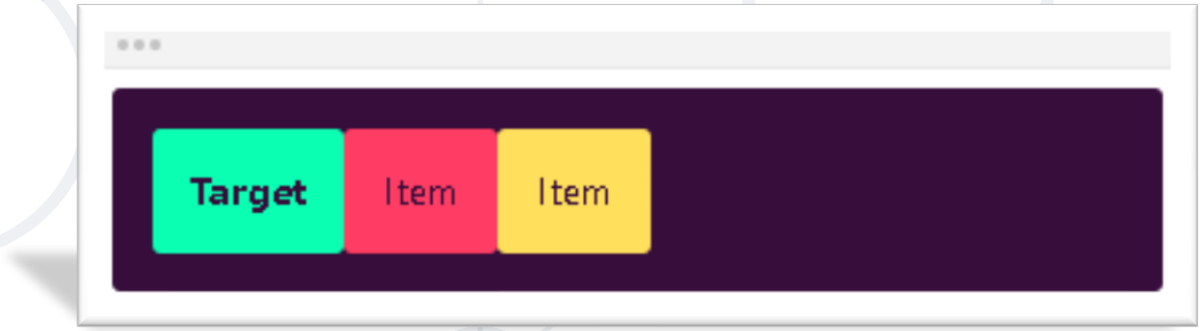
You can set a **different** value for each flexbox item



Flex-grow

- Flex-grow - defines how much a flexbox item should **grow** if there's space available
  - **flex-grow: 0;**

  The element will **not** grow if there's space available. It will only use the space it needs



Flex-grow

- **flex-grow: 1;**

The element will **grow** by a factor of 1. It will fill up the remaining space if no other flexbox item has a flex-grow value

Flex-shrink

- Flex-shrink - defines how much a flexbox item should **shrink** if there's **not enough** space available
  - **flex-shrink: 1;**

If there's **not enough** space available in the container's main axis, the element will **shrink** by a factor of **1**, and will wrap its content

Flex-shrink

- **flex-shrink: 0;**

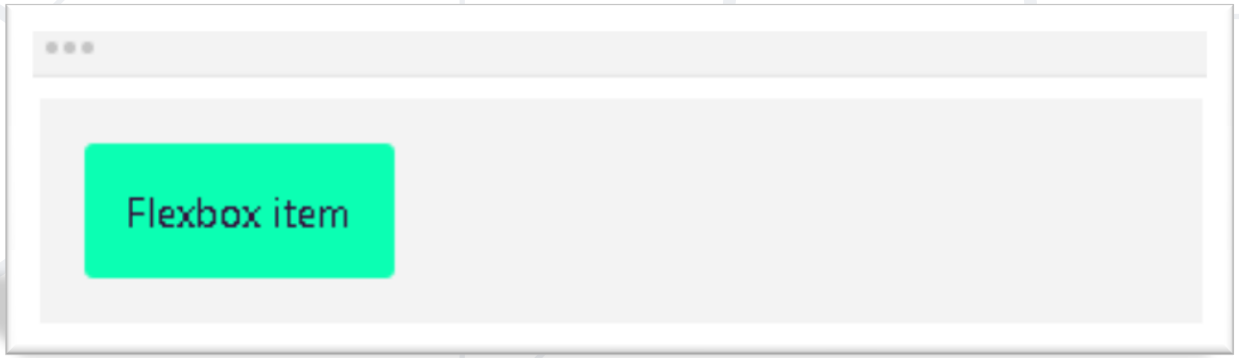
The element will **not** shrink it will retain the width it needs, and **not** wrap its content. Its siblings will shrink to give space to the target element.

Because the target element will not wrap its content, there is a chance for the flexbox container's content to **overflow**

Flex-basis

- Flex-basis - defines the initial size of a flexbox item
  - **flex-basis: auto;**

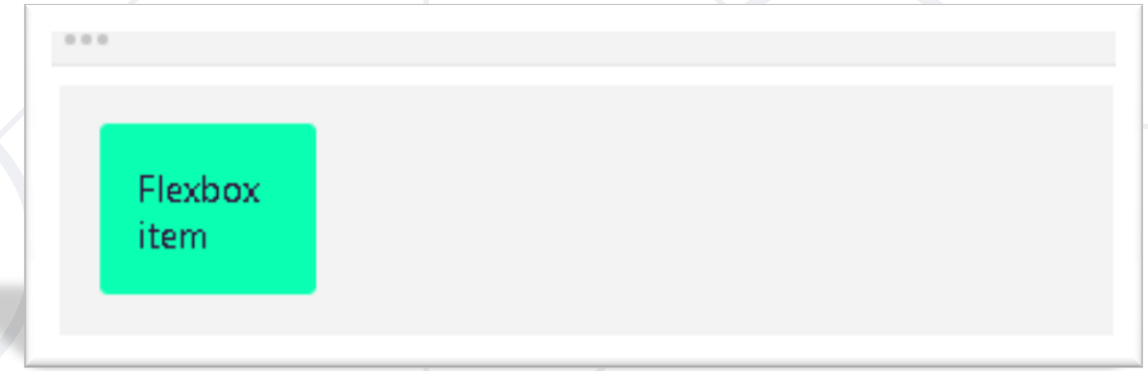
  The element will be automatically sized based on its content, or on any height or width value if they are defined



Flex-basis

- **flex-basis: 80px;**

You can define **pixel** or **(r)em** values. The element will wrap its content to avoid any overflow



Flex

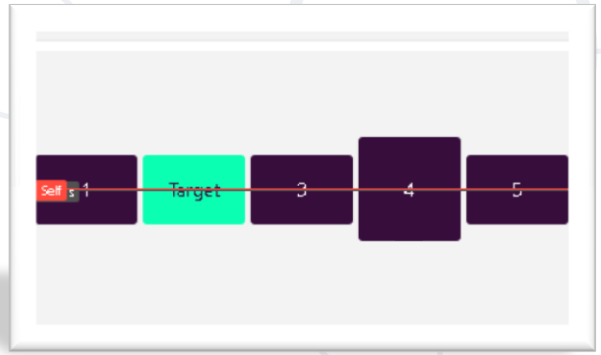
- Flex is the shorthand for:
  - flex-grow
  - flex-shrink
  - flex-basis
- The default is **0 1 auto**

```
.item {
      flex: <flex-grow> <flex-shrink> <flex-basis>
}
```

Align-self

- Align-self – works like align-items, but applies only to a **single** flexbox item, instead of all of them
  - **align-self: auto;**

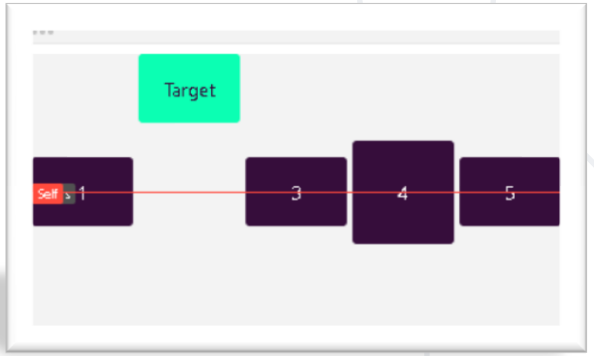
The target will use the value of align-items



  - **align-self: flex-start;**

The container has align-items: center

The target has align-self: flex-start

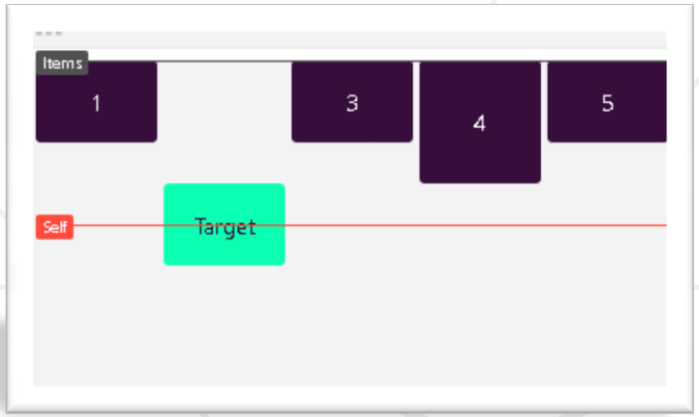


Align-self

- **align-self: center;**

The container has align-items: flex-start

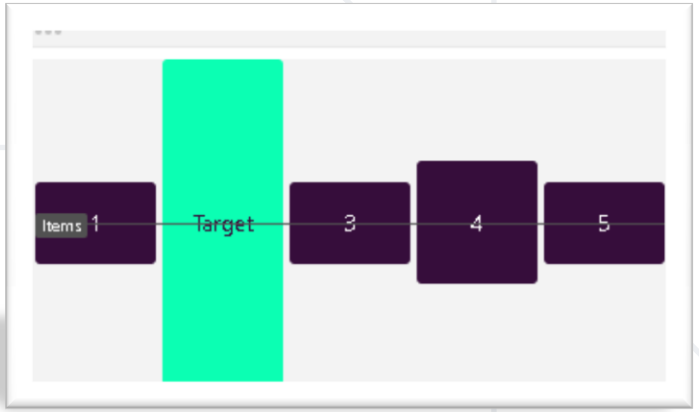
The target has align-self: center



- **align-self: stretch;**

The container has align-items: center

The target has align-self: stretch

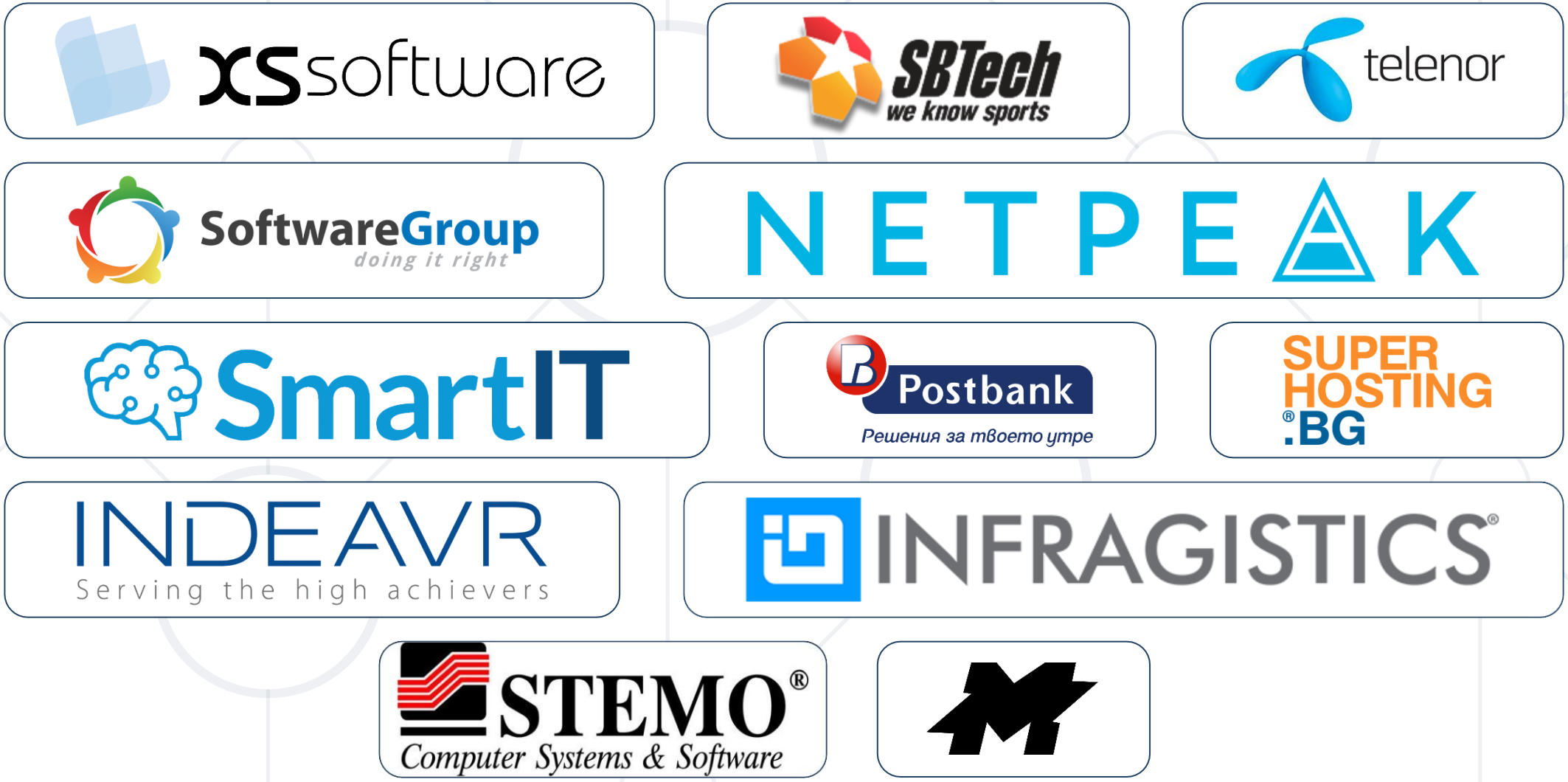


Summary

- What is Flexbox?
- Why Flexbox?
- Properties for the Parent: display, directio n, wrap, justify, align
- Properties for the children: order, shrink, align

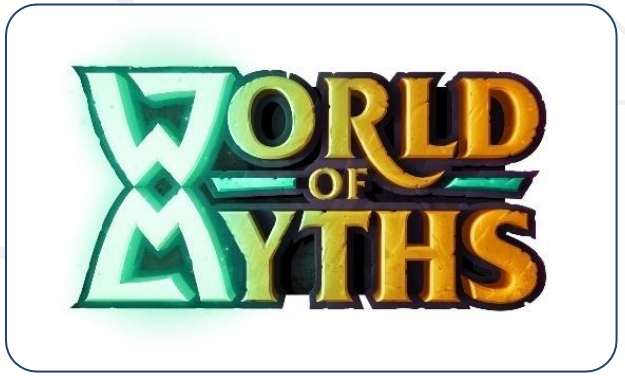
SoftUni Diamond Partners

Software University

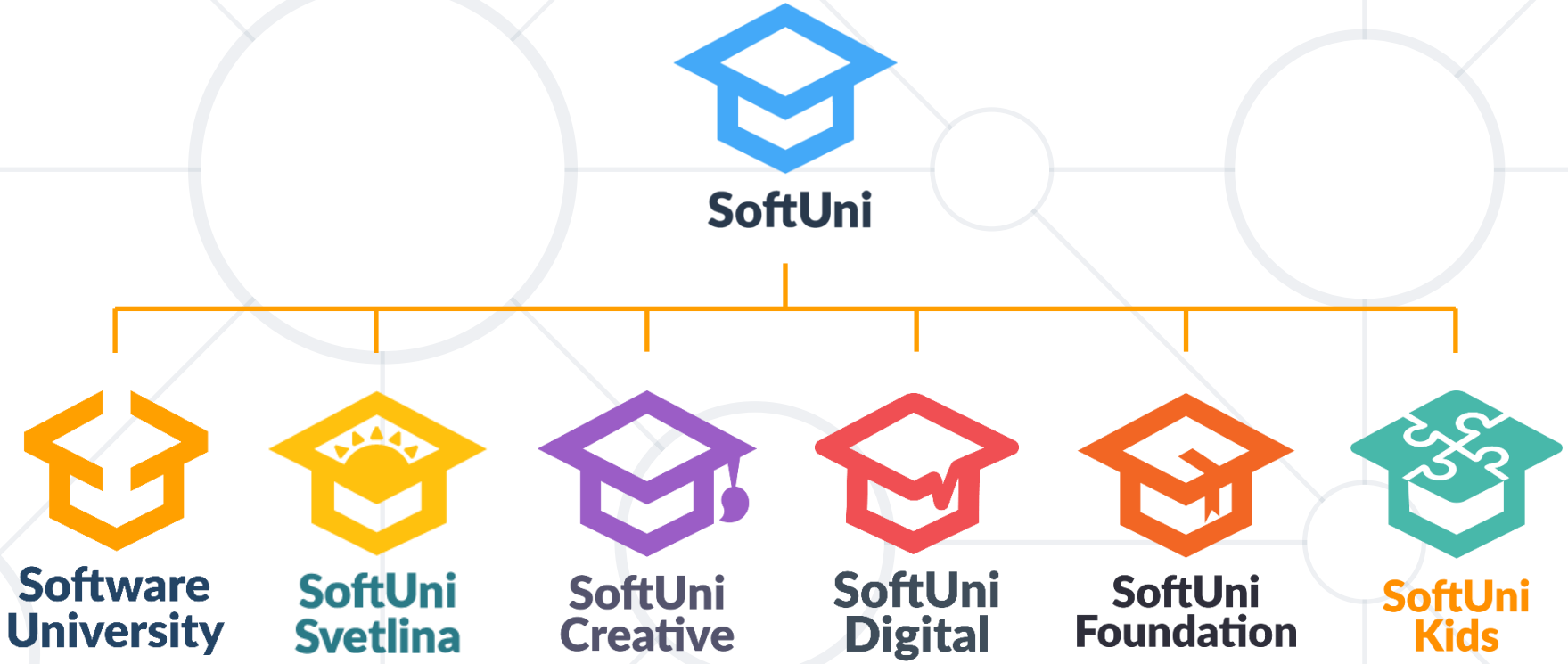


SoftUni Organizational Partners

Software University



Questions?



License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://softuni.org
- © Software University – https://softuni.bg

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, softuni.org
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg